FIGURE 1

```
contend():

s ← 0;

repeat [

    Q₁ ← ∅;

    ‖_{u∈U} [ statusᵤ ← u,try();

        Q₁ ← {u} ∪ Q₁;

    ] until (∃Q ∈ Q:Q ⊆ Q₁);

    if (|{u : statusᵤ = LOCKED}|≤b)

        return;

    else

        s ← s + 1;

        d ← ᵣ[(Δ+4δ)]... 2ˢ(Δ+4δ);

        sleep(d);

    ] until (false);
```

$$s \leftarrow 0;$$

$$Q_1 \leftarrow \emptyset;$$

$$\|_{u \in U} [\ status_u \leftarrow u, try();$$

$$Q_1 \leftarrow \{u\} \cup Q_1;$$

$$]\ until\ (\exists Q \in \mathcal{Q}: Q \subseteq Q_1);$$

$$if\ (|\{u : status_u = LOCKED\}| \leq b)$$

$$s \leftarrow s + 1;$$

$$d \leftarrow {}_R[(\Delta+4\delta)]\ldots 2^s(\Delta+4\delta);$$

FIGURE 2A: MUTUAL EXCLUSION PROTOCOL CLIENT PROGRAM

```
try():

    if (clock() - lastGranted > Δ + 2 δ)

        lastGranted ← clock();

        return FREE;

    else

        return LOCKED;
```

FIGURE 2B: MUTUAL EXCLUSION PROTOCOL SERVER PROGRAM

```
1)   submit(o):
2)   waiting ← true;
3)   ‖ V₁ ← ∅; Q₁ ←_R Q;
4)      ‖_{u∈Q1} [p_u ← u.submit(o);V₁←
          {p_u}∪ V₁;
5)      ] until (∃p:|{p_u ∈ V₁
      :p=p_u}|≥b+1;
6)      waiting ← false;
7)      return p:| {p_u ∈ V₁:p=p_u}|
          ≥b+1
8)   ‖ repeat [
9)       contend();
10)      Q₂ ← ∅;
11)      ‖_{u∈U} [⟨σ_u^c, σ_u^pc,
      proposer_u,, pending_u⟩←u.get(r);
12)          Q₂←{u} ∪ Q₂;
13)      ] until (∃Q ∈ Q:Q ⊆ Q₂);
14)      Σ^c←{σ':|{u:σ'=σ_u^c   }|≥b+1};
15)      σ^c←:σ:σ.version=max_{σ'∈Σc}
          {σ'.version};
16)      σ^pc←choose({⟨σ_u^pc,
      proposer_u⟩:σ  v≠⊥});
17)      completed←max
          {completed,max{v:|{u:σ_u^pc
          .version>v|≥b+1}};
18)      if σ^c≠⊥ ∧ σ^c.version>
          completed)
19)          σ←σ^c
20)      else if (σ^pc≠⊥ ∧ σ^pc.
      version > completed)
21)          σ←σ^pc;
22)      else
23)          pending←{o:∈{u:o ∈
          pending_u}|≥b+1};
24)          σ ← apply  (pending
      σ^c);
25)      Q₂ ← ∅;
26)      ‖_{u∈U} [u.propose(σ' r);Q₂←
          {u}∪Q₂;
27)      ]until (∃Q∈Q:Q⊆Q₂;

28)      Q₂ ← ∅;

29)      ‖_{u∈U} [u.commit(σ,r);Q₂ ←
          {u}∪Q₂;
30)      ]until(∃Q∈Q:Q⊆Q₂);

31)      completed ← max
      {completed,σ.version};

32)   ] until (waiting =
      false);

33) choose ({⟨σ_u^pc,proposer_u⟩}u):
34)     S[1,2,...]←{⟨σ_u^pc ,
    proposer_u⟩}_u sorted in
descending order by ⟨σ,
proposer⟩ > ⟨σ',
proposer'⟩↔(σ.version >
σ'.version ∨
(σ.version =
σ'.version ∧ proposer    >
proposer'});
35)     i ← 1; count ←
[0,0,...];
36)     repeat
        [⟨σ,proposer⟩←S[i];
37)         count[σ]←count[σ]+1;
38)         i ← i + 1;
39)     ] until(∃σ:count[σ]
        ≥b+1 ∨ i > |S|)'
40)     if (∃σ:count[σ]≥b+1
41)         return σ:count
            [σ]≥b+1;
42)     else
43)         return ⊥;
44) apply (pending, σ'):
45)     repeat [ o ←_R pending;
46)         pending←pending\{o};
47)         if(σ'.reflects
            (o)=false)
48)             σ'.doOp(o);
49)     ] until (pending=∅)'
50)     σ'.version←σ'.
        version+1'
51)     return σ';
```

FIGURE 3A: CLIENT SIDE OF AN ORDERING PROTOCOL

```
1)   submit (o):                    5)   get(r):
2)       pending←pending ∪{o};      6)       if ® > maxRank)
3)       sleep until                7)           maxRank ← r;
     (response(o)≠⊥);               8)           return ⟨◻⊥◻σᶜ,σᵖᶜ,
4)       return response (o);                     prosper, pending⟩
                                    9)       else
                                    10)          throw RankException;

11)  propose (σ,r):                 19)  commit (σ,r):
12)      if ® ≥ maxRank)            20)      if ® ≥ maxRank)
13)          maxRank ← r;           21)          maxRank ← r;
14)          proposer ← r;          22)          σᶜ, σ;
15)          σᵖᶜ ← σ;               23)          pending ← pending \
16)          return;                             {o:σ.reflects(o)=true};
17)      else                       24)          response ← response|σ.
18)          throw RankException;                response;
                                    25)          return;
                                    26)      else
                                    27)          throw RankException;
```

FIGURE 3B: SERVER SIDE OF AN ORDERING PROTOCOL